Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre

Lab Work
“Lab-5”

[Code No: COMP-342]

Submitted by:

Bibhushan Saakha [41]

Submitted to:

Mr Dhiraj Shrestha
Department of Computer Science and Engineering

Submission Date:
2024-06-13

Questions:

1. Implement Liang Barsky Line Clipping algorithm.
2. Implement Sutherland Hodgeman polygon clipping algorithm.

1. Liang Barsky Line Clipping Algorithm

The Liang–Barsky algorithm uses the parametric equation of a line and inequalities describing the range of the clipping window to determine the intersections between the line and the clip window. With these intersections it knows which portion of the line should be drawn.

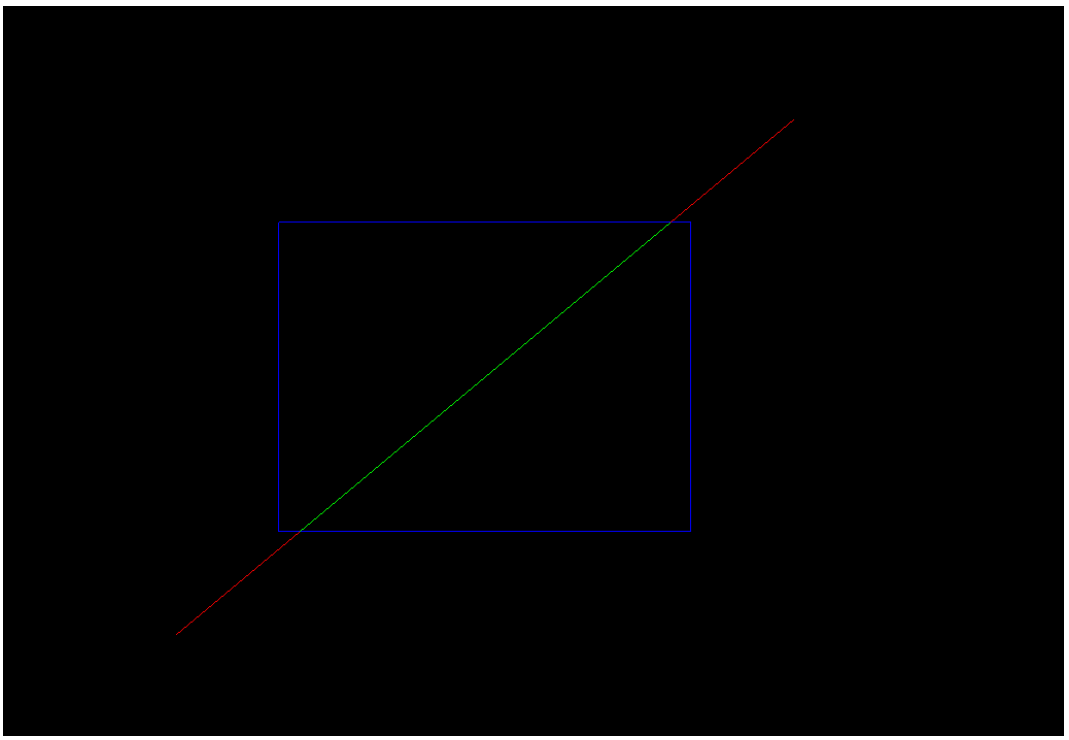
Code:

```python
import glfw
from OpenGL.GL import *
import numpy as np

def init_window(width, height):
    if not glfw.init():
        return None

    window = glfw.create_window(width, height, "Liang-Barsky Line Clipping", None,
None)if not window:
        glfw.terminate()
        return None

    glfw.make_context_current(window)
    glOrtho(-width // 2, width // 2, -height // 2, height // 2, -1, 1)
    return window

def plot_line(x0, y0, x1, y1, color):
    glColor3f(*color)
    glBegin(GL_LINES)
    glVertex2f(x0, y0)
    glVertex2f(x1, y1)
    glEnd()

def liang_barsky(x0, y0, x1, y1, xmin, ymin, xmax, ymax):
    dx, dy = x1 - x0, y1 - y0
    p = [-dx, dx, -dy, dy]
    q = [x0 - xmin, xmax - x0, y0 - ymin, ymax - y0]

    u1, u2 = 0.0, 1.0

    for i in range(4):
        if p[i] == 0 and q[i] < 0:
            return None
        if p[i] < 0:
            u1 = max(u1, q[i] / p[i])
        elif p[i] > 0:
            u2 = min(u2, q[i] / p[i])

    if u1 > u2:
        return None

    x0_clipped = x0 + u1 * dx
    y0_clipped = y0 + u1 * dy
    x1_clipped = x0 + u2 * dx
    y1_clipped = y0 + u2 * dy

    return x0_clipped, y0_clipped, x1_clipped, y1_clipped

```

```
1
2 def main():
3     window = init_window(800, 600)
4     if not window:
5         return
6
7     x0, y0 = -100, -100
8     x1, y1 = 200, 150
9     xmin, ymin = -50, -50
10    xmax, ymax = 150, 100
11
12    while not glfw.window_should_close(window):
13        glClear(GL_COLOR_BUFFER_BIT)
14
15        # Plot clipping window
16        glColor3f(0.0, 0.0, 1.0)  # Blue
17        glBegin(GL_LINE_LOOP)
18        glVertex2f(xmin, ymin)
19        glVertex2f(xmax, ymin)
20        glVertex2f(xmax, ymax)
21        glVertex2f(xmin, ymax)
22        glEnd()
23
24        # Plot original line
25        plot_line(x0, y0, x1, y1, (1.0, 0.0, 0.0))  # Red
26
27        # Plot clipped line
28        clipped_line = liang_barsky(x0, y0, x1, y1, xmin, ymin, xmax,
29 ymax)    if clipped_line:
30            plot_line(*clipped_line, (0.0, 1.0, 0.0))  # Green
31
32        glfw.swap_buffers(window)
33        glfw.poll_events()
34
35    glfw.terminate()
36
37 if __name__ == "__main__":
38     main()
39
```

This Python script implements the Sutherland-Hodgman polygon clipping algorithm using OpenGL and GLFW. It initializes a window with orthographic projection, defines functions to plot polygons and compute intersections, and applies the clipping algorithm to clip a polygon against a specified clipping window. The main loop continuously updates the display to show the original and clipped polygons until the window is closed.

Output:



2. Sutherland Hodgeman Polygon Clipping Algorithm

The Sutherland–Hodgman Algorithm works by extending each line of the convex clip polygon in turn and selecting only vertices from the subject polygon that are on the visible side.

Code:

```python
import glfw
from OpenGL.GL import *
from OpenGL.GLU import *
import numpy as np

win_width, win_height = 800, 800

original_polygon = [(100, 150), (200, 250), (300, 200), (350, 150), (250, 100)]
clip_rect = [150, 100, 300, 250]  # [xmin, ymin, xmax, ymax]

def sutherland_hodgman_clip(polygon, clip_rect):
    def clip_polygon(poly, edge):
        clipped_polygon = []
        x0, y0 = poly[-1]
        for x1, y1 in poly:
            if inside(x1, y1, edge):
                if not inside(x0, y0, edge):
                    clipped_polygon.append(intersect(x0, y0, x1, y1, edge))
                clipped_polygon.append((x1, y1))
            elif inside(x0, y0, edge):
                clipped_polygon.append(intersect(x0, y0, x1, y1, edge))
            x0, y0 = x1, y1
        return clipped_polygon

    def inside(x, y, edge):
        if edge == 'left':
            return x >= xmin
        elif edge == 'right':
            return x <= xmax
        elif edge == 'bottom':
            return y >= ymin
        elif edge == 'top':
            return y <= ymax

    def intersect(x0, y0, x1, y1, edge):
        if edge == 'left':
            x = xmin
            y = y0 + (xmin - x0) * (y1 - y0) / (x1 - x0)
        elif edge == 'right':
            x = xmax
            y = y0 + (xmax - x0) * (y1 - y0) / (x1 - x0)
        elif edge == 'bottom':
            y = ymin
            x = x0 + (ymin - y0) * (x1 - x0) / (y1 - y0)
        elif edge == 'top':
            y = ymax
            x = x0 + (ymax - y0) * (x1 - x0) / (y1 - y0)
        return (x, y)

    xmin, ymin, xmax, ymax = clip_rect
    edges = ['left', 'right', 'bottom', 'top']
    output_polygon = polygon
    for edge in edges:
        output_polygon = clip_polygon(output_polygon, edge)
    return output_polygon

```

```python
def draw_polygon(polygon):
    glBegin(GL_POLYGON)
    for x, y in polygon:
        glVertex2f(x, y)
    glEnd()

def draw_clipping_region():
    xmin, ymin, xmax, ymax = clip_rect
    glColor3f(0, 1, 0)  # Green color
    glBegin(GL_LINE_LOOP)
    glVertex2f(xmin, ymin)
    glVertex2f(xmax, ymin)
    glVertex2f(xmax, ymax)
    glVertex2f(xmin, ymax)
    glEnd()

def main():
    if not glfw.init():
        return

    window = glfw.create_window(win_width, win_height, "Sutherland-Hodgman Polygon Clipping", None,
None)if not window:
        glfw.terminate()
        return

    glfw.make_context_current(window)

    glViewport(0, 0, win_width, win_height)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluOrtho2D(0, win_width, 0, win_height)

    while not glfw.window_should_close(window):
        glClear(GL_COLOR_BUFFER_BIT)

        draw_clipping_region()

        glColor3f(1, 0, 0)  # Red color
        draw_polygon(original_polygon)

        glColor3f(0, 0, 1)  # Blue color
        clipped_polygon = sutherland_hodgman_clip(original_polygon, clip_rect)
        if clipped_polygon:
            draw_polygon(clipped_polygon)

        glfw.swap_buffers(window)
        glfw.poll_events()

    glfw.terminate()

if __name__ == "__main__":
    main()

```

This Python script implements the Sutherland-Hodgman polygon clipping algorithm using OpenGL and GLFW. It initializes an 800x800 window with orthographic projection, defines functions for polygon clipping against a rectangular clipping region, and continuously updates the display to show both the original and clipped polygons until the window is closed. The clipping process is visually represented in green for the clipping region, red for the original polygon, and blue for the clipped polygon.

Output: